Supplementary Material for Variance Reduction and Quasi-Newton for Particle-Based Variational Inference

Michael H. Zhu [1] **Chang Liu** [2] **Jun Zhu** [3]

Algorithm A.1 SPIDER for ParVIs

Require: Initial particles $\{x_0^{(j)}\}_{j=1}^M$, target distribution $p_0(x)\prod_{n=1}^N p_n(x)$, update period T_s, learning rate ε.

Require: Vector field estimators $\hat{U}(\{x^{(j)}\}_j)^{(i)}$ and $\hat{V}_n(\{x^{(j)}\}_j)^{(i)}$, parallel transport estimator $\hat{\Gamma}_{\{x^{(j)}\}_j}^{\{y^{(j)}\}_j}\left(\{V^{(j)}\}_j\right)^{(i)}$.

Initialize $\tilde{x}^{(i)} \leftarrow x_0^{(i)}$ for $i = 1, \cdots, M$.
for $s = 1, 2, 3, \cdots$ **do**
  Let $x_0^{(i)} \leftarrow \tilde{x}^{(i)}$ for $i = 1, \cdots, M$.
  for $k = 0, \cdots, T_s - 1$ **do**
    if $k = 0$ **then**
      Let $W_0^{(i)} \leftarrow \hat{U}(\{x_0^{(j)}\}_j)^{(i)} + \sum_{n=1}^N \hat{V}_n(\{x_0^{(j)}\}_j)^{(i)}$ for $i = 1, \cdots, M$.
    else
      Choose a data point $n_k \in \{1, \cdots, N\}$ uniformly at random.
      Let $W_k^{(i)} \leftarrow \hat{U}(\{x_k^{(j)}\}_j)^{(i)} + N\hat{V}_{n_k}(\{x_k^{(j)}\}_j)^{(i)} - \hat{\Gamma}_{\{x_{k-1}^{(j)}\}_j}^{\{x_k^{(j)}\}_j}\left(\left\{\hat{U}(\{x_{k-1}^{(j')}\}_{j'})^{(j)} + N\hat{V}_{n_k}(\{x_{k-1}^{(j')}\}_{j'})^{(j)} - W_{k-1}^{(j)}\right\}_j\right)^{(i)}$ for $i = 1, \cdots, M$.
    end if
    Let $x_{k+1}^{(i)} \leftarrow x_k^{(i)} + \varepsilon W_k^{(i)}/\|W_k\|$, where $\|W_k\|^2 = \frac{1}{M}\sum_{j=1}^M \|W_k^{(j)}\|^2$, for $i = 1, \cdots, M$.
  end for
  Let $\tilde{x}^{(i)} \leftarrow x_{T_s}^{(i)}$ for $i = 1, \cdots, M$.
end for
return $\{\tilde{x}^{(i)}\}_{i=1}^M$.

[1]Department of Computer Science, Stanford University, Stanford, CA, USA [2]Microsoft Research Asia, Beijing, 100080, China [3]Dept. of Comp. Sci. & Tech., Institute for AI, BNRist Center, Tsinghua-Bosch ML Center, Tsinghua University, Beijing, 100084, China. Correspondence to: J. Zhu <dcszj@tsinghua.edu.cn>, Chang Liu <changliu@microsoft.com>, Michael H. Zhu <mhzhu@cs.stanford.edu>.

Proceedings of the 37^{th} International Conference on Machine Learning, Online, PMLR 119, 2020.

Algorithm A.2 L-BFGS two-loop recursion in SQN-VR for ParVIs

Require: Memory size τ, L-BFGS pairs $\left\{\left([S_u^{(j)}]_{j=1}^M, [Y_u^{(j)}]_{j=1}^M\right)\right\}_{u=s-\tau+1}^{s}$, gradient $[W^{(j)}]_{j=1}^M$.

Let $Q^{(i)} \leftarrow W^{(i)}$ for $i = 1, \cdots, M$.
for $u = s, s-1, \cdots, s-\tau+1$ **do**
  Let $\rho_u \leftarrow 1/\sum_{j=1}^M \left(S_u^{(j)}\right)^T Y_u^{(j)}$.
  Let $\alpha_u \leftarrow \rho_u \sum_{j=1}^M \left(S_u^{(j)}\right)^T Q^{(j)}$.
  Let $Q^{(i)} \leftarrow Q^{(i)} - \alpha_u Y_u^{(i)}$ for $i = 1, \cdots, M$.
end for
Let $Z^{(i)} \leftarrow \frac{\sum_{j=1}^M \left(S_s^{(j)}\right)^T Y_s^{(j)}}{\sum_{j=1}^M \left(Y_s^{(j)}\right)^T Y_s^{(j)}} Q^{(i)}$ for $i = 1, \cdots, M$.
for $u = s-\tau+1, s-\tau+2, \cdots, s$ **do**
  Let $\beta \leftarrow \rho_u \sum_{j=1}^M \left(Y_u^{(j)}\right)^T Z^{(j)}$.
  Let $Z^{(i)} \leftarrow Z^{(i)} + (\alpha_u - \beta) S_u^{(i)}$ for $i = 1, \cdots, M$.
end for
return $\{Z^{(i)}\}_{i=1}^M$.

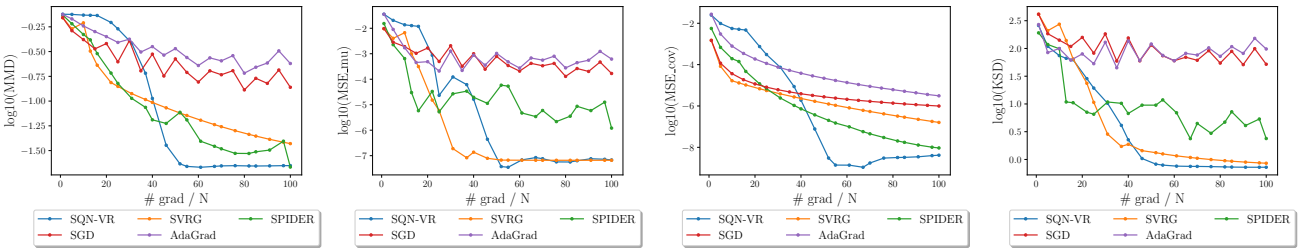
Algorithm A.3 Stochastic Quasi-Newton with Variance Reduction (SQN-VR) for ParVIs

Require: Initial particles $\{x_0^{(j)}\}_{j=1}^M$, target distribution $p_0(x)\prod_{n=1}^N p_n(x)$, number of epochs S, update period T_s, learning rates $\varepsilon_1, \varepsilon_2$, L-BFGS memory size, cautious update threshold c.

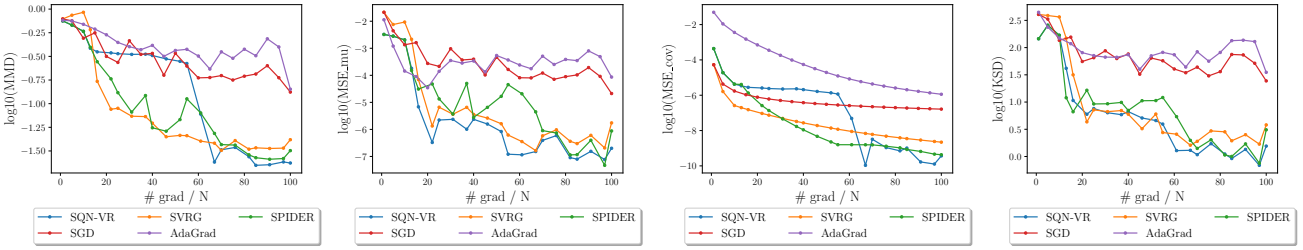
Require: Vector field estimators $\hat{U}(\{x^{(j)}\}_j)^{(i)}$ and $\hat{V}_n(\{x^{(j)}\}_j)^{(i)}$, parallel transport estimator $\hat{\Gamma}^{\{y^{(j)}\}_j}_{\{x^{(j)}\}_j}\left(\{V^{(j)}\}_j\right)^{(i)}$, inverse exponential map estimator $\widehat{\mathrm{Exp}}^{-1}_{\{x^{(j)}\}_j}(\{y^{(j)}\}_j)^{(i)}$.

Initialize $\tilde{x}_1^{(i)} \leftarrow x_0^{(i)}$ for $i = 1, \cdots, M$.
Let $\tilde{V}_1^{(i)} \leftarrow \sum_{n=1}^N \hat{V}_n(\{\tilde{x}_1^{(j)}\}_j)^{(i)}$ for $i = 1, \cdots, M$.
for $s = 1, 2, 3, \cdots, S$ **do**
  Let $x_0^{(i)} \leftarrow \tilde{x}_s^{(i)}$ for $i = 1, \cdots, M$.
  for $k = 0, \cdots, T_s - 1$ **do**
    Choose a data point $n_k \in \{1, \cdots, N\}$ uniformly at random.
    if $s > 2$ **then**
      Let $W_k^{(i)} = \left(\hat{\Gamma}^{\{x_k^{(j)}\}_j}_{\{\tilde{x}_s^{(j)}\}_j}\right)^{-1}\left(\left\{\hat{U}(\{x_k^{(j')}\}_{j'})^{(j)} + N\hat{V}_{n_k}(\{x_k^{(j')}\}_{j'})^{(j)}\right\}_j\right)^{(i)} - \left(N\hat{V}_{n_k}(\{\tilde{x}_s^{(j')}\}_{j'})^{(i)} - \tilde{V}_s^{(i)}\right)$ for $i = 1, \cdots, M$.
      Compute the quasi-Newton update $[Z_k^{(j)}]_{j=1}^M$ from $[W_k^{(j)}]_{j=1}^M$ by L-BFGS two-loop recursion.
      Let $x_{k+1}^{(i)} \leftarrow x_k^{(i)} - \varepsilon_2 \hat{\Gamma}^{\{x_k^{(j)}\}_j}_{\{\tilde{x}_s^{(j)}\}_j}\left(\{Z_k^{(j)}\}_j\right)^{(i)}$ for $i = 1, \cdots, M$.
    else
      Let $W_k^{(i)} = \hat{U}(\{x_k^{(j)}\}_j)^{(i)} + N\hat{V}_{n_k}(\{x_k^{(j)}\}_j)^{(i)} - \hat{\Gamma}^{\{x_k^{(j)}\}_j}_{\{\tilde{x}_s^{(j)}\}_j}\left(\left\{N\hat{V}_{n_k}(\{\tilde{x}_s^{(j')}\}_{j'})^{(j)} - \tilde{V}_s^{(j)}\right\}_j\right)^{(i)}$ for $i = 1, \cdots, M$.
      Let $x_{k+1}^{(i)} \leftarrow x_k^{(i)} + \varepsilon_1 W_k^{(i)}$ for $i = 1, \cdots, M$.
    end if
  end for
  Let $\tilde{x}_{s+1}^{(i)} \leftarrow x_{T_s}^{(i)}$ for $i = 1, \cdots, M$.
  Let $\tilde{V}_{s+1}^{(i)} \leftarrow \sum_{n=1}^N \hat{V}_n(\{\tilde{x}_{s+1}^{(j)}\}_j)^{(i)}$ for $i = 1, \cdots, M$.
  Calculate the tangent vector $\eta^{(i)}$ from $\tilde{x}_s^{(i)}$ to $\tilde{x}_{s+1}^{(i)}$ by $\eta^{(i)} = \widehat{\mathrm{Exp}}^{-1}_{\{\tilde{x}_s^{(j)}\}_j}(\{\tilde{x}_{s+1}^{(j)}\}_j)^{(i)}$ for $i = 1, \cdots, M$.
  Let $S_{s+1}^{(i)} \leftarrow \hat{\Gamma}^{\{\tilde{x}_{s+1}^{(j)}\}_j}_{\{\tilde{x}_s^{(j)}\}_j}\left(\left\{\eta^{(j)}\right\}_j\right)^{(i)}$ for $i = 1, \cdots, M$.
  Let $Y_{s+1}^{(i)} \leftarrow \hat{U}(\{\tilde{x}_{s+1}^{(j)}\}_j)^{(i)} + \tilde{V}_{s+1}^{(i)} - \hat{\Gamma}^{\{\tilde{x}_{s+1}^{(j)}\}_j}_{\{\tilde{x}_s^{(j)}\}_j}\left(\left\{\hat{U}(\{\tilde{x}_s^{(j')}\}_{j'})^{(j)} + \tilde{V}_s^{(j)}\right\}_j\right)^{(i)}$ for $i = 1, \cdots, M$.
  if $\sum_{j=1}^M \left(S_{s+1}^{(j)}\right)^T Y_{s+1}^{(j)} \geq c\sum_{j=1}^M \|S_{s+1}^{(j)}\|^2$ **then**
    Store the L-BFGS pair $\left([S_{s+1}^{(j)}]_{j=1}^M, [Y_{s+1}^{(j)}]_{j=1}^M\right)$, and discard the oldest pair if the memory size is exceeded.
  end if
  Transport the L-BFGS pairs $\left\{\left([S_{s'}^{(j)}]_{j=1}^M, [Y_{s'}^{(j)}]_{j=1}^M\right)\right\}_{s'<s+1}$ from $\{\tilde{x}_s^{(j)}\}_j$ to $\{\tilde{x}_{s+1}^{(j)}\}_j$ using $\hat{\Gamma}^{\{\tilde{x}_{s+1}^{(j)}\}_j}_{\{\tilde{x}_s^{(j)}\}_j}$.
end for
return $\{\tilde{x}_{S+1}^{(i)}\}_{i=1}^M$.

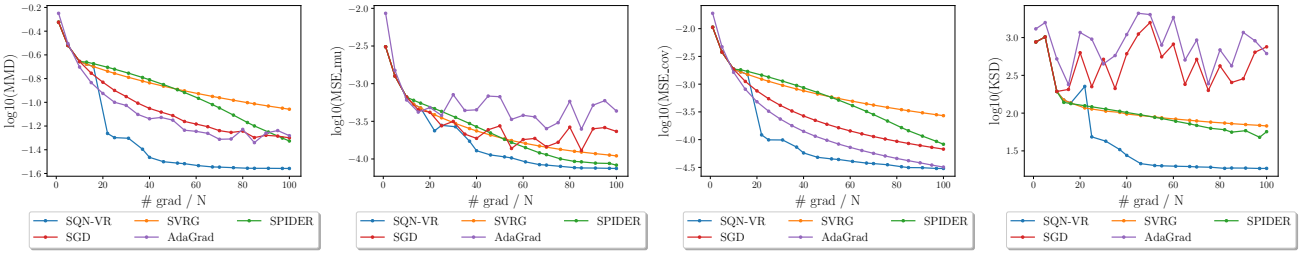
Experimental results on all datasets



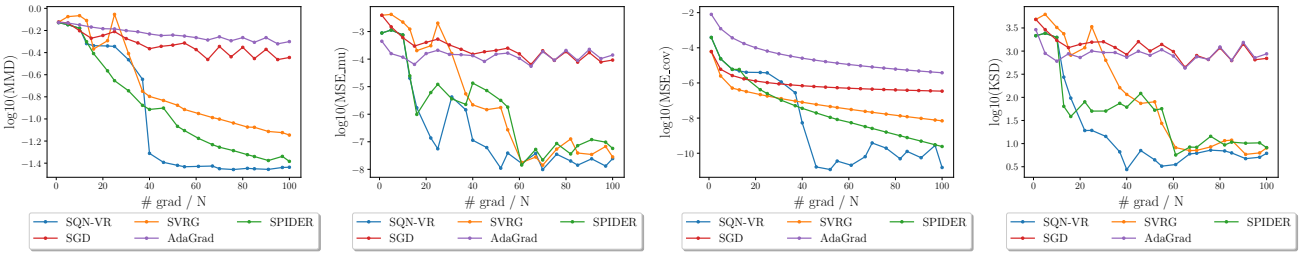
(a) N = 1030, D = 9, cond(Σ) = 73, concrete



(b) N = 1503, D = 6, cond(Σ) = 12, noise



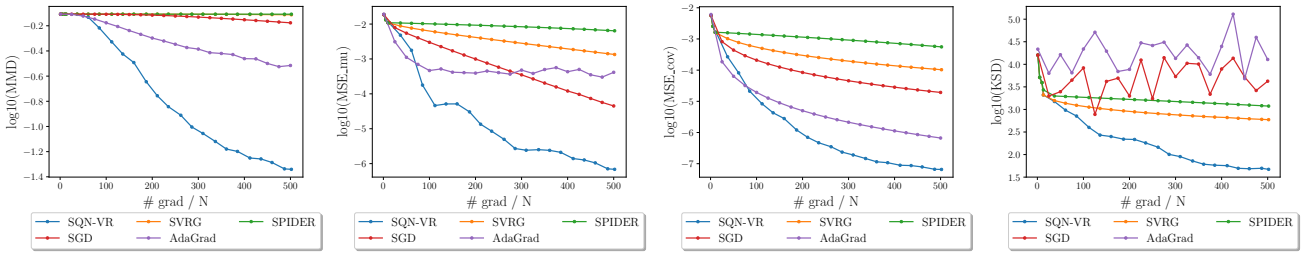
(c) N = 5875, D = 21, cond(Σ) = 65697, parkinson



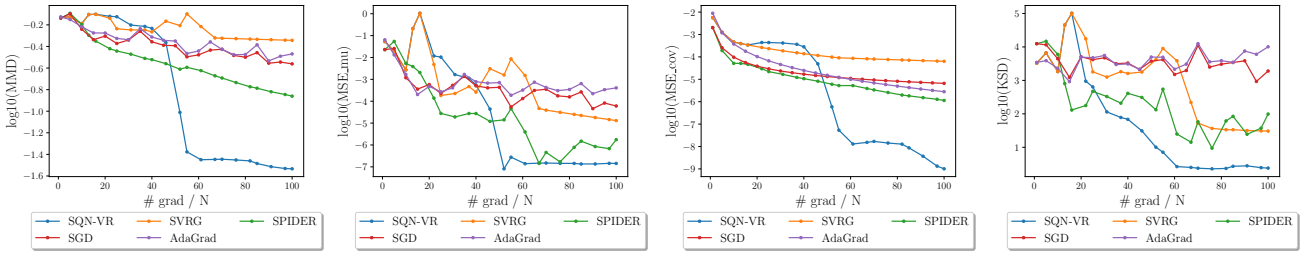
(d) N = 17379, D = 13, cond(Σ) = 218, bike

Figure A.1. Experimental results for Bayesian linear regression (page 1)

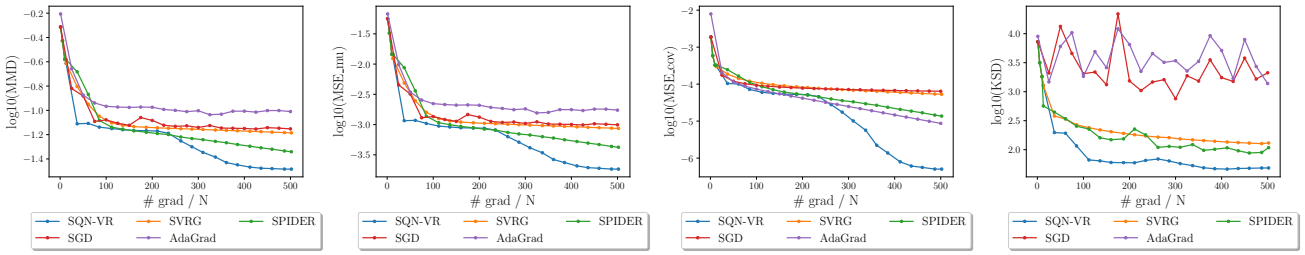
Experimental results on all datasets



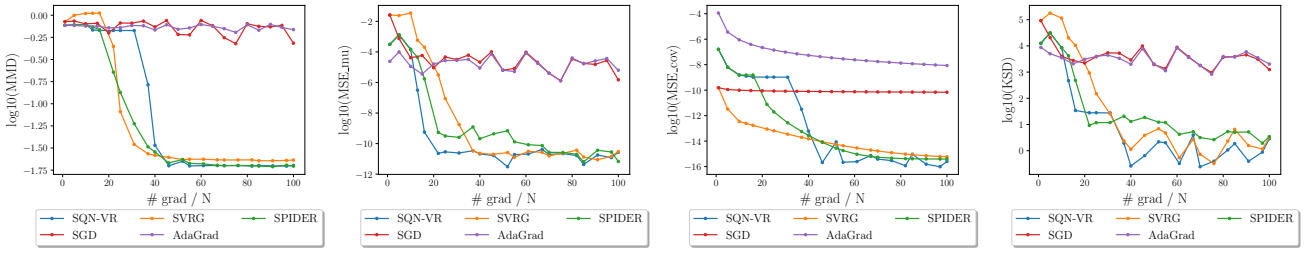
(e) N = 28179, D = 97, cond(Σ) = 45923, toms



(f) N = 45730, D = 10, cond(Σ) = 4527, protein



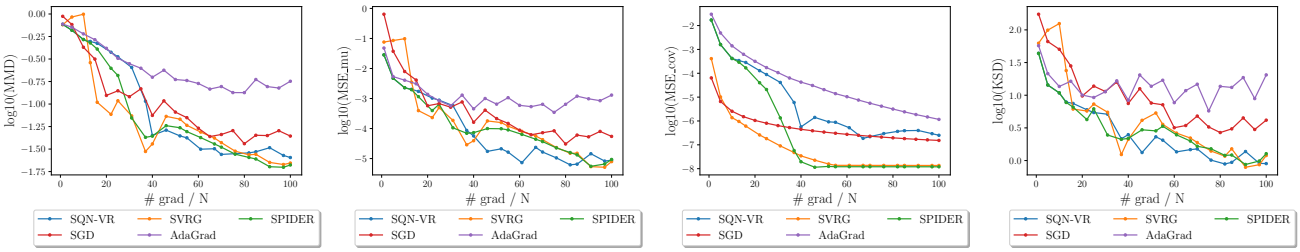
(g) N = 64608, D = 28, cond(Σ) = 701531, kegg



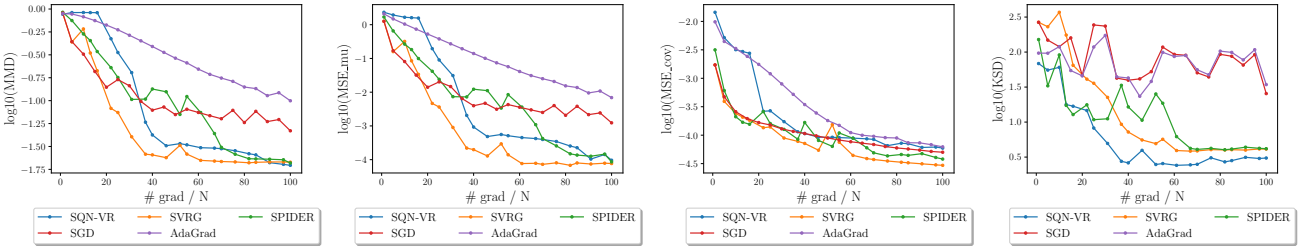
(h) N = 434874, D = 3, cond(Σ) = 4, 3droad

Figure A.2. Experimental results for Bayesian linear regression (page 2)

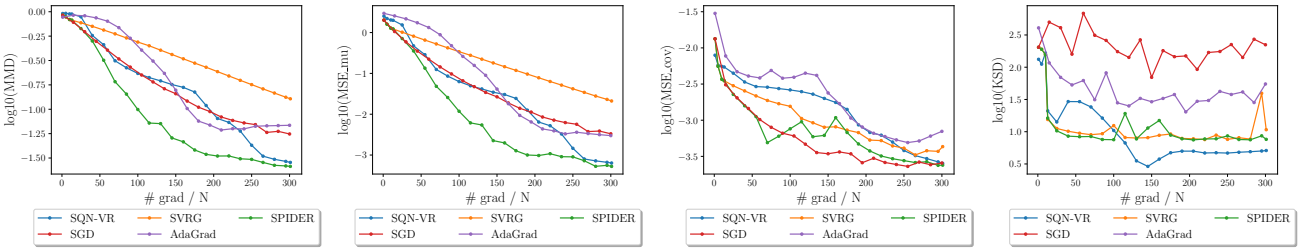
Experimental results on all datasets



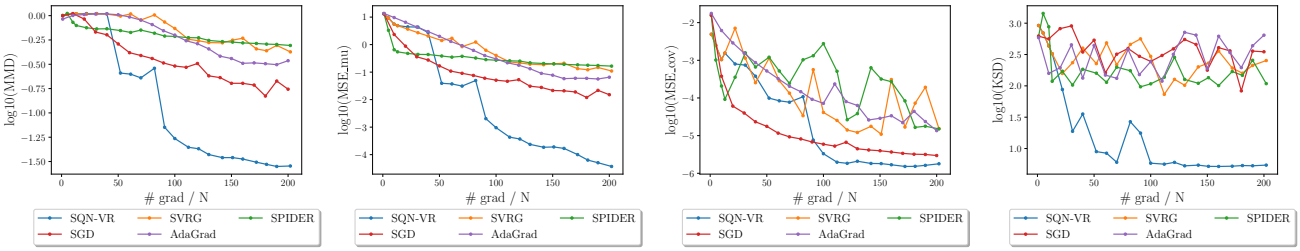
(a) N = 768, D = 9, cond(Σ) = 5, pima



(b) N = 1151, D = 20, cond(Σ) = 1106, diabetic



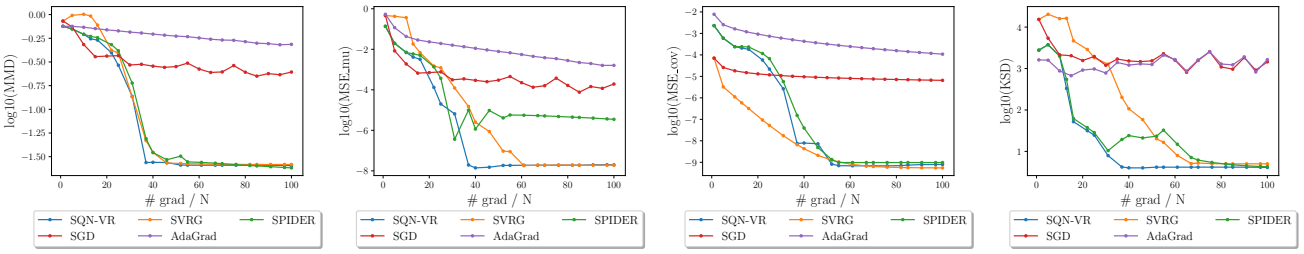
(c) N = 14980, D = 15, cond(Σ) = 4730, eeg



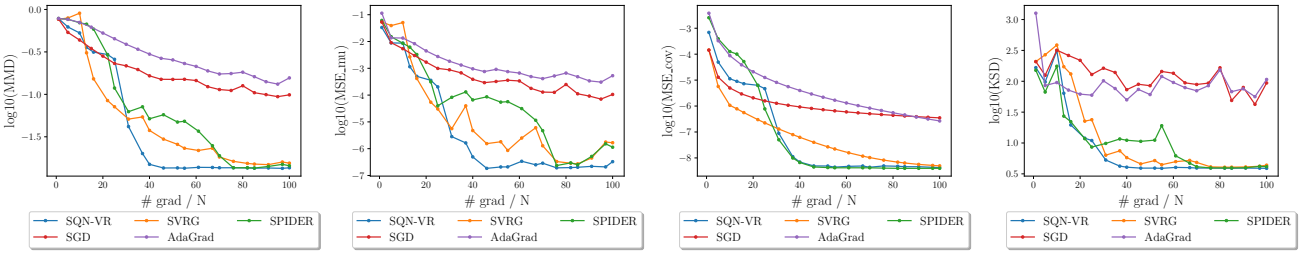
(d) N = 58000, D = 10, cond(Σ) = 3109, space

Figure A.3. Experimental results for Bayesian logistic regression (page 1)

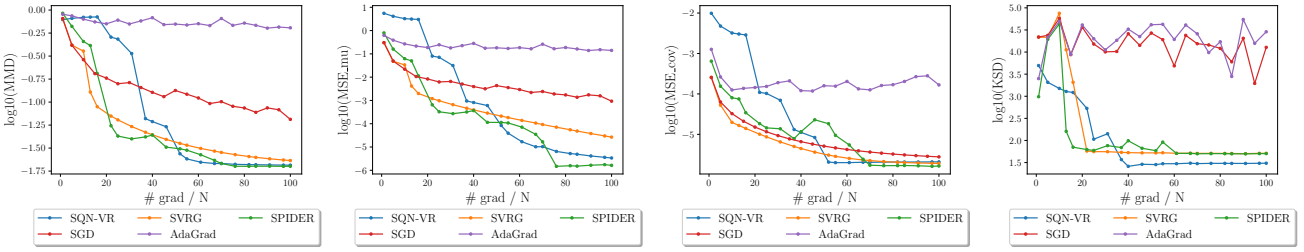
Experimental results on all datasets



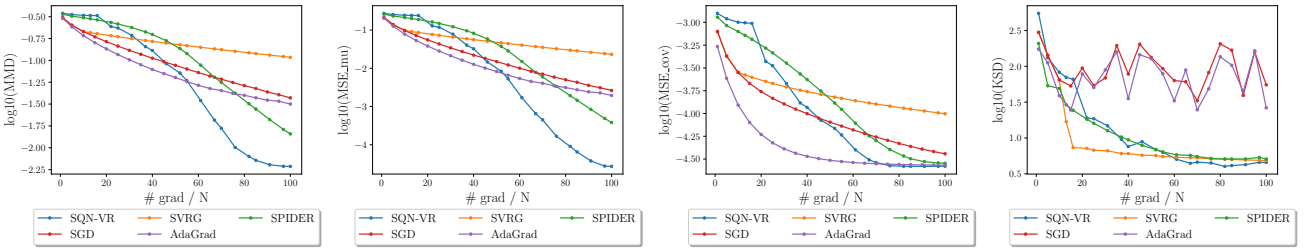
(e) N = 100000, D = 19, cond(Σ) = 2412, susy



(f) N = 12214, D = 51, cond(Σ) = 58, mnist (with reduced dimension of 50 using PCA, same as in the main paper)



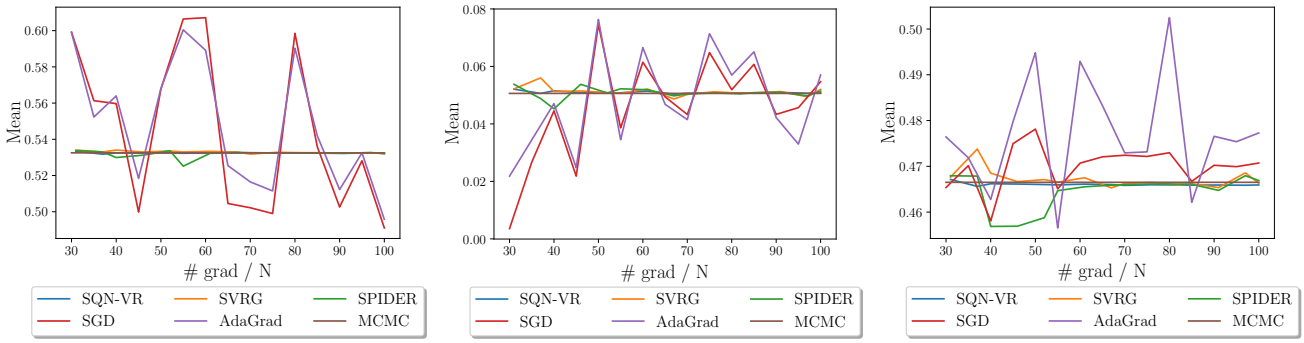
(g) N = 464809, D = 55, cond(Σ) = 341266, covtype



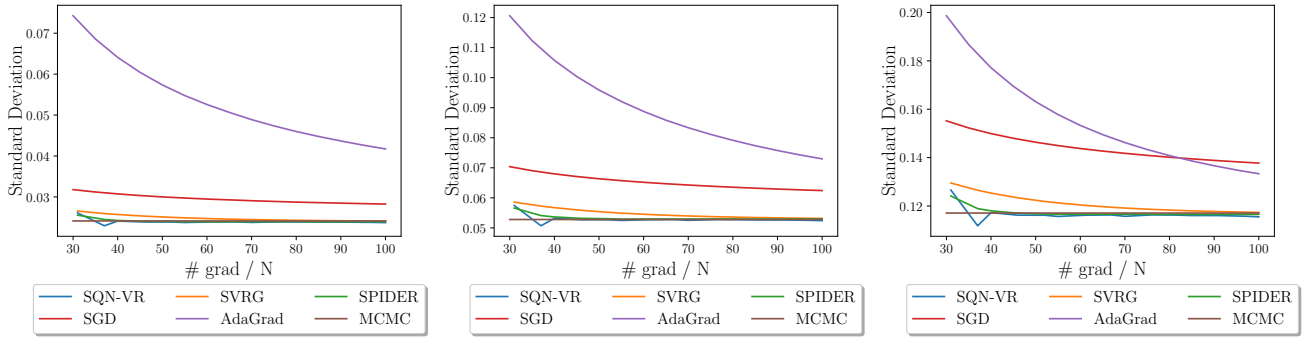
(h) N = 12214, D = 785, mnist with 28×28 image pixels as input (using 1,000 particles)

Figure A.4. Experimental results for Bayesian logistic regression (page 2)

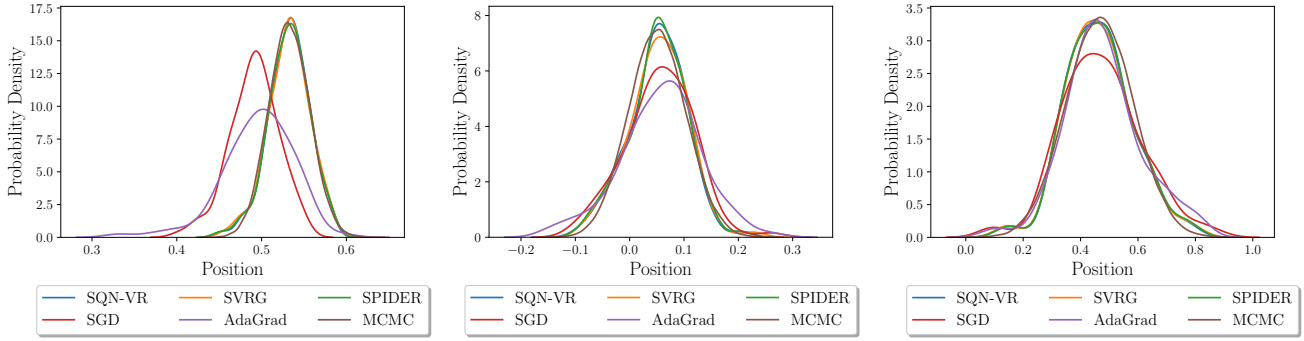
Visualizations



(a) Mean of the particles as a function of number of iterations



(b) Standard deviation of the particles as a function of number of iterations



(c) Estimated marginal posterior distribution using kernel density estimation

Figure A.5. Visualizations for 3 selected dimensions out of 51 on the mnist dataset. The 3 columns represent the 3 selected dimensions. For each dimension and each algorithm, (a) the mean and (b) the standard deviation of the particles are plotted as a function of the number of iterations, and (c) the kernel density estimate of the final set of particles is shown. We see that the methods with variance reduction (SQN-VR, SVRG, SPIDER) achieve much better approximation of the posterior.

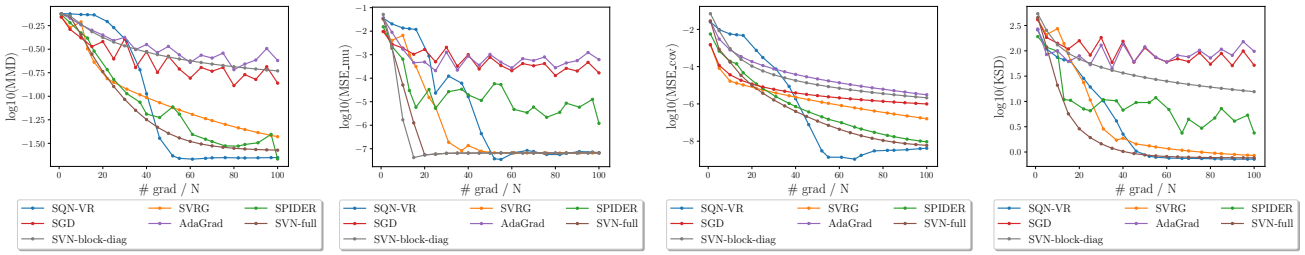
Comparison with Stein variational Newton

We compare with the Stein variational Newton (SVN) method by Detommaso et al. (2018). We implemented two versions of SVN, "full" and "block-diagonal" approximation. For SVN, we use the entire dataset for computing the gradient and Hessian since there is no extension of SVN to the mini-batch setting. We choose the number of iterations for SVN so that the total number of passes over the dataset for SVN is the same as the other methods. The experimental results are presented in Figures A.6 to A.9.

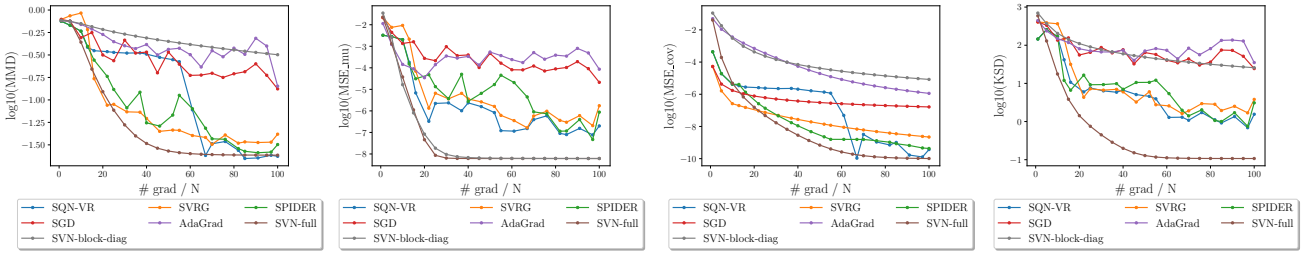
SVN-full performs comparable to SQN-VR in many of the experiments. The experiments with the biggest differences in terms of log10(MMD) are SVN-full being better than SQN-VR by about 0.4 on toms and kegg but being worse than SQN-VR by about 1.3 on mnist and 0.9 on covtype. However, the running time of SVN-full can be much longer than SQN-VR. For example, our implementation of SVN-full took around 5.5 hours for toms (vs 15 minutes for SQN-VR) and 30 minutes for kegg (vs 10 minutes for SQN-VR).

SVN-block-diagonal, in terms of MMD, is comparable to SQN-VR on 2 of the 8 linear regression datasets but is worse than SQN-VR on all of the other datasets. In these cases, SVN-block-diagonal is often comparable to SGD and AdaGrad in terms of MMD.

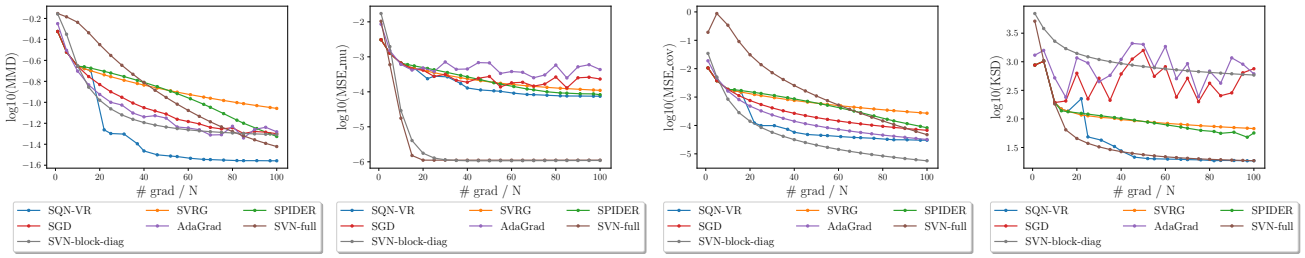
Comparison with Stein variational Newton



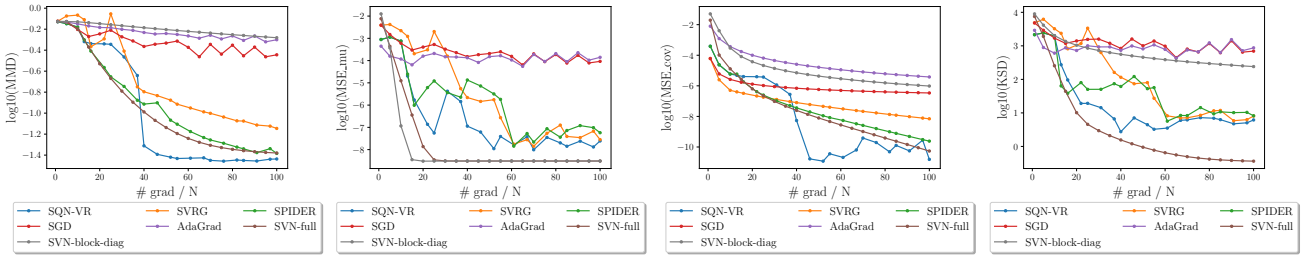
(a) N = 1030, D = 9, cond(Σ) = 73, concrete



(b) N = 1503, D = 6, cond(Σ) = 12, noise



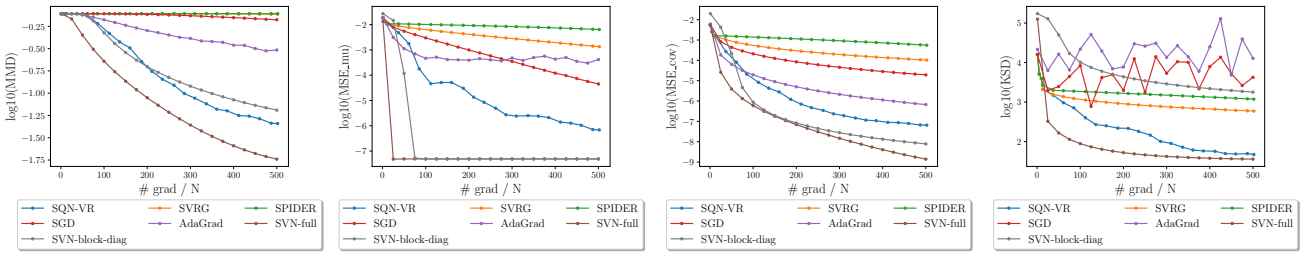
(c) N = 5875, D = 21, cond(Σ) = 65697, parkinson



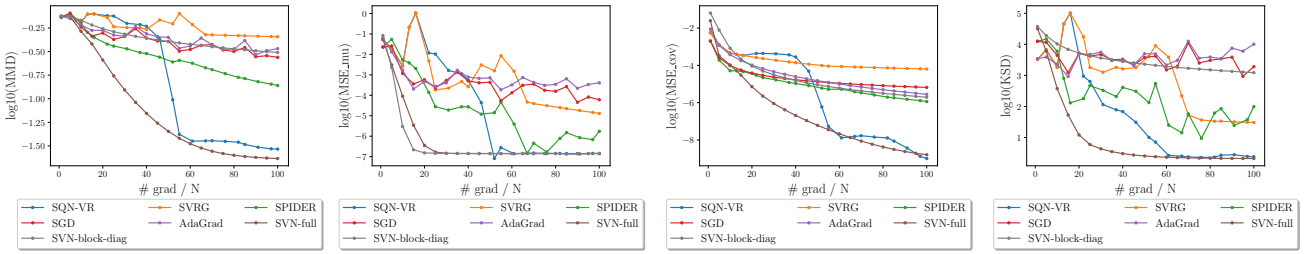
(d) N = 17379, D = 13, cond(Σ) = 218, bike

Figure A.6. Comparison with Stein variational Newton for Bayesian linear regression (page 1)

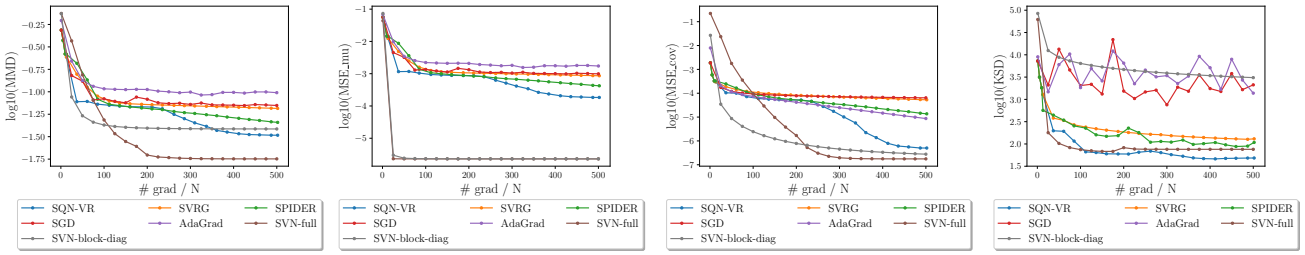
Comparison with Stein variational Newton



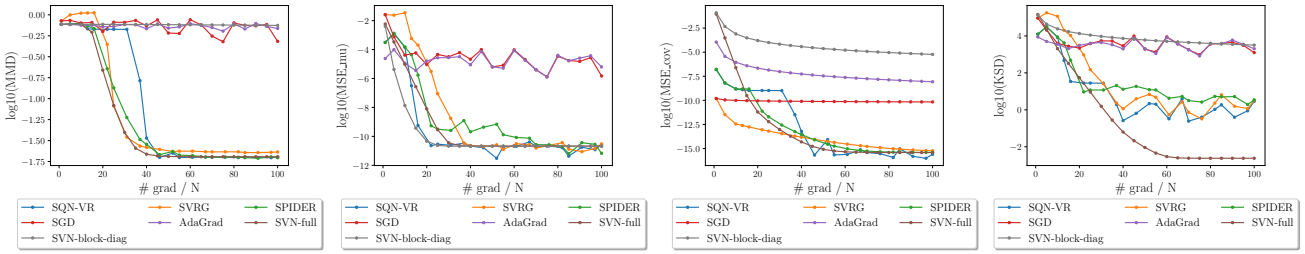
(e) N = 28179, D = 97, cond(Σ) = 45923, toms



(f) N = 45730, D = 10, cond(Σ) = 4527, protein



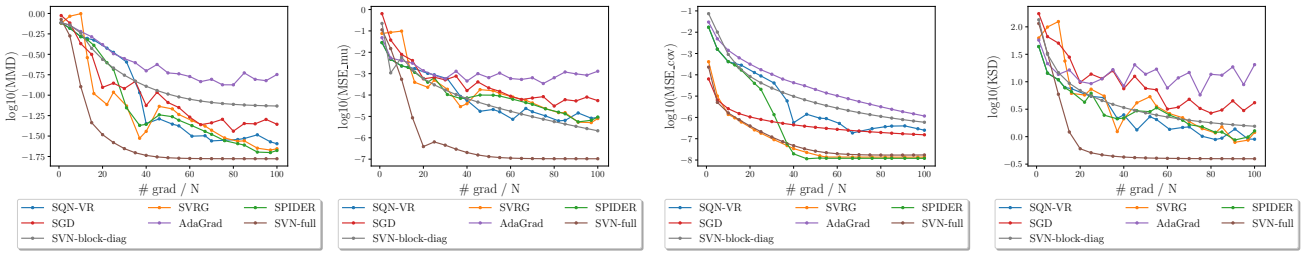
(g) N = 64608, D = 28, cond(Σ) = 701531, kegg



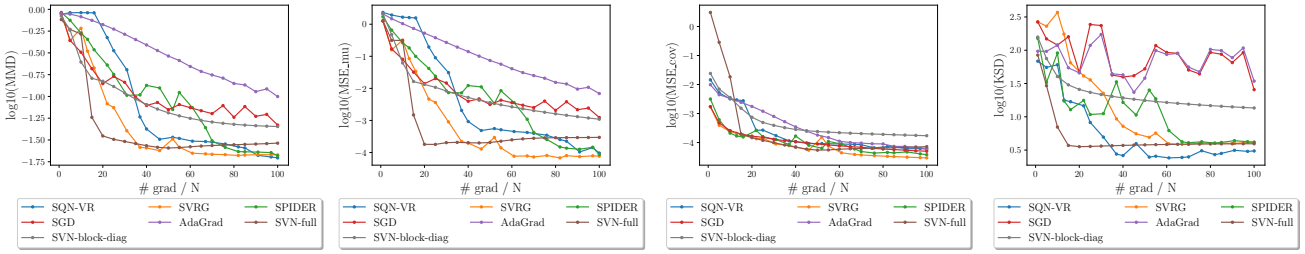
(h) N = 434874, D = 3, cond(Σ) = 4, 3droad

Figure A.7. Comparison with Stein variational Newton for Bayesian linear regression (page 2)

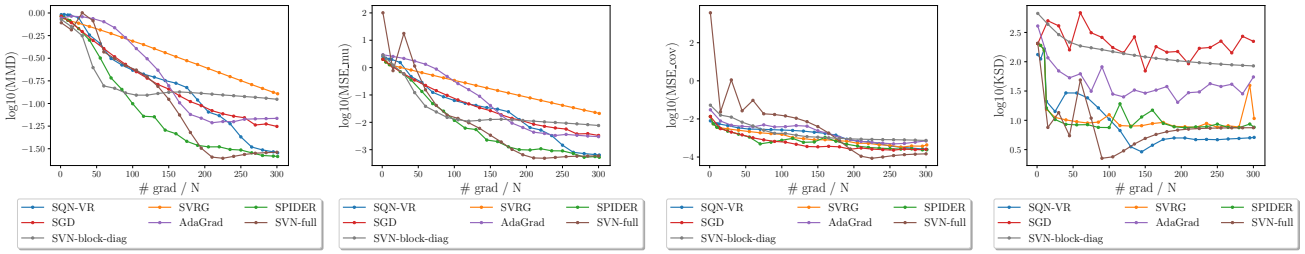
Comparison with Stein variational Newton



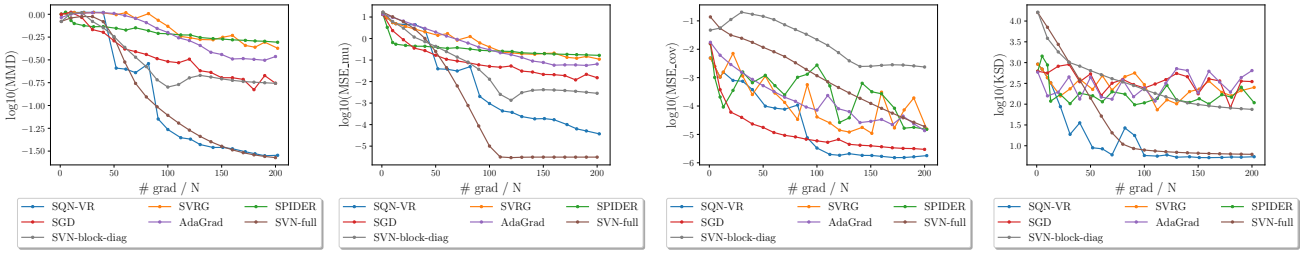
(a) N = 768, D = 9, cond(Σ) = 5, pima



(b) N = 1151, D = 20, cond(Σ) = 1106, diabetic



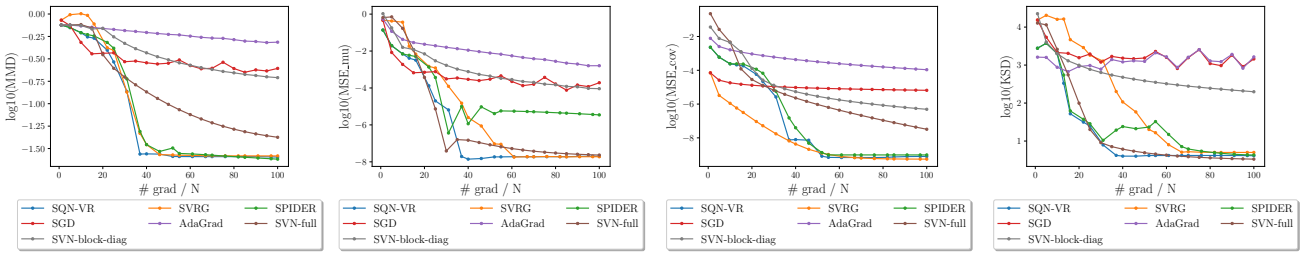
(c) N = 14980, D = 15, cond(Σ) = 4730, eeg



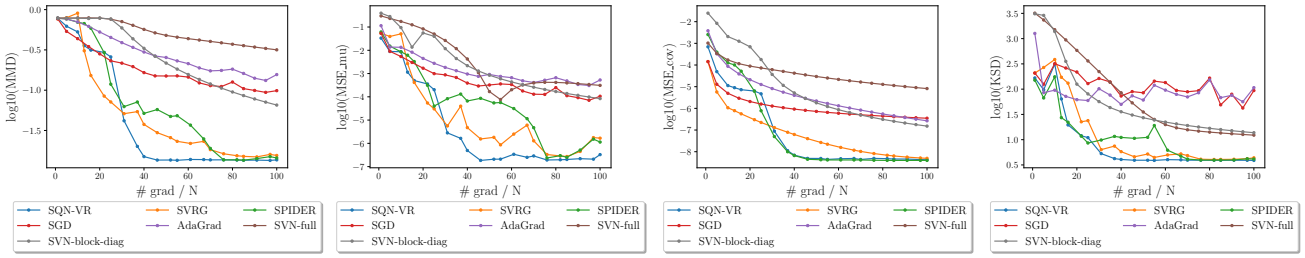
(d) N = 58000, D = 10, cond(Σ) = 3109, space

Figure A.8. Comparison with Stein variational Newton for Bayesian logistic regression (page 1)

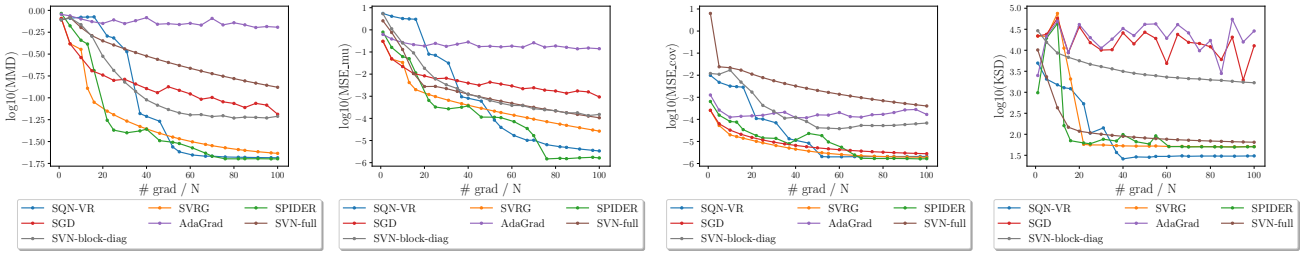
Comparison with Stein variational Newton



(e) N = 100000, D = 19, cond(Σ) = 2412, susy



(f) N = 12214, D = 51, cond(Σ) = 58, mnist



(g) N = 464809, D = 55, cond(Σ) = 341266, covtype

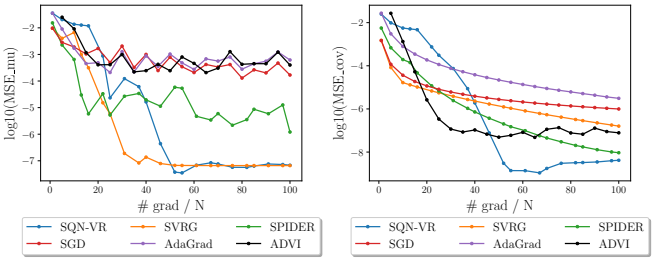
Figure A.9. Comparison with Stein variational Newton for Bayesian logistic regression (page 2)

Comparison with ADVI

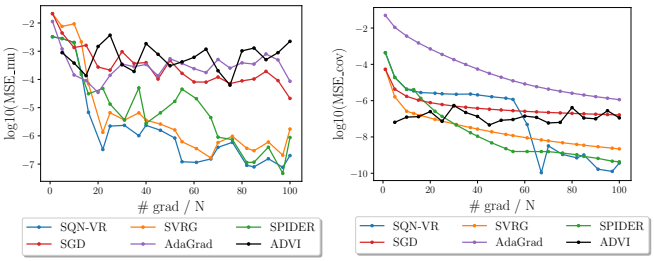
We compare with Automatic Differentiation Variational Inference (ADVI) by Kucukelbir et al. (2017). Specifically, we run Gaussian parametric variational inference with a full-rank Gaussian using the implementation of ADVI in PyMC3 (https://docs.pymc.io/). We use a similar experimental setup as in our paper: a batch size of 10, 100 Monte Carlo samples from the variational Gaussian distribution to estimate the loss (corresponding to our 100 particles), and AdaGrad optimizer. We do a grid search over learning rates and choose the one that attains the lowest loss at the end of the run. The experimental results are shown in Figures A.10 to A.13.

We find that the ADVI solution has much higher MSE_mu and MSE_cov than SQN-VR on every dataset except toms, for which ADVI has lower MSE_cov by 10^1 but higher MSE_mu by 10^3, and was often comparable to AdaGrad. One possible explanation is the doubly stochastic (data sub-sampling and Monte Carlo sampling from the variational distribution) gradients in ADVI, so ADVI only learns an approximate solution.

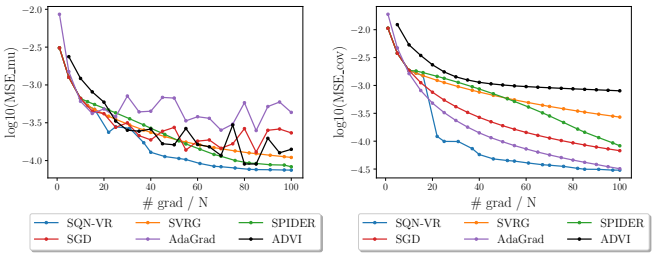
Comparison with ADVI



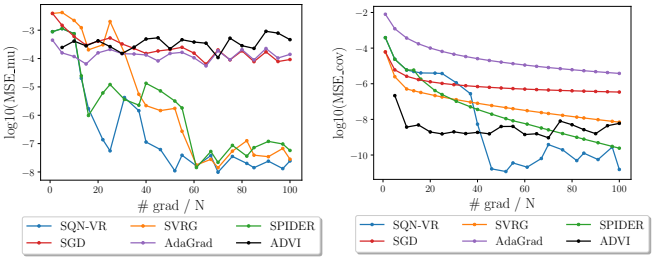
(a) N = 1030, D = 9, cond(Σ) = 73, concrete



(b) N = 1503, D = 6, cond(Σ) = 12, noise



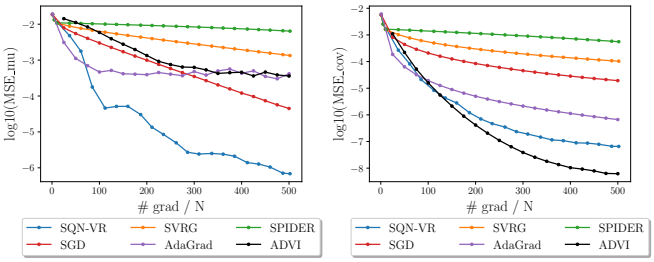
(c) N = 5875, D = 21, cond(Σ) = 65697, parkinson



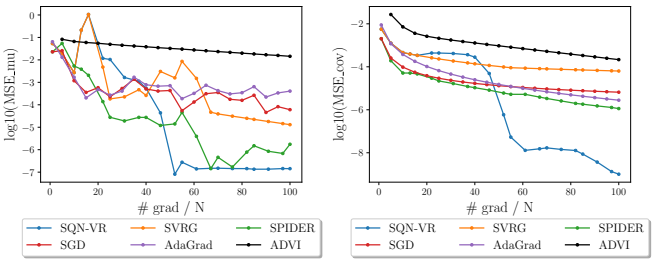
(d) N = 17379, D = 13, cond(Σ) = 218, bike

Figure A.10. Comparison with ADVI for Bayesian linear regression (page 1)

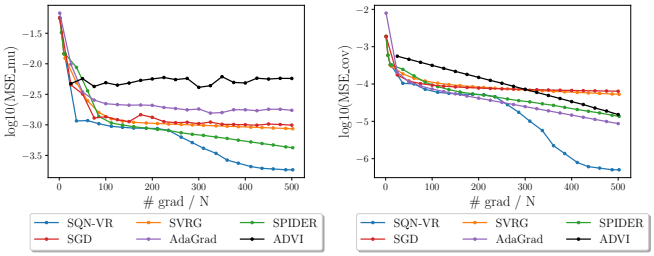
Comparison with ADVI



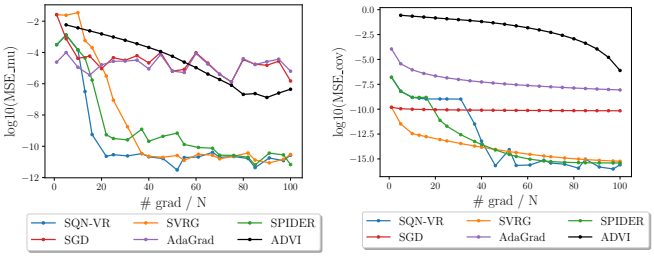
(e) N = 28179, D = 97, cond(Σ) = 45923, toms



(f) N = 45730, D = 10, cond(Σ) = 4527, protein



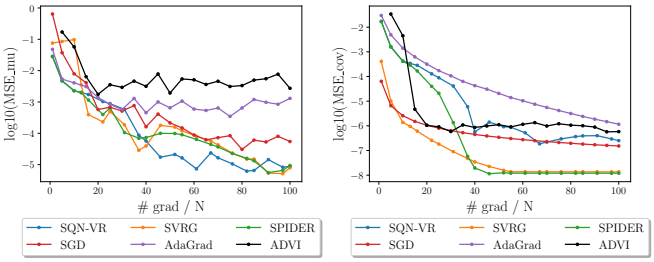
(g) N = 64608, D = 28, cond(Σ) = 701531, kegg



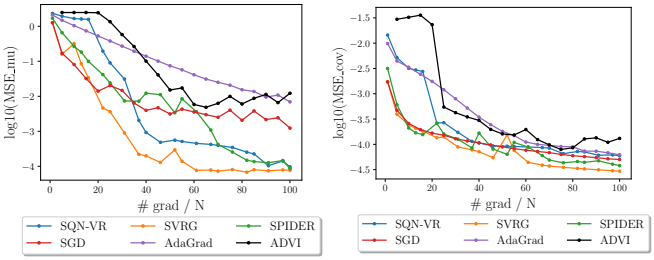
(h) N = 434874, D = 3, cond(Σ) = 4, 3droad

Figure A.11. Comparison with ADVI for Bayesian linear regression (page 2)

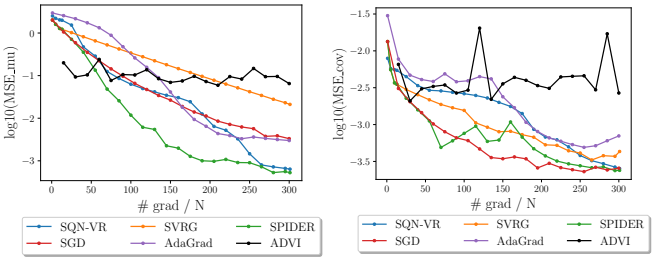
Comparison with ADVI



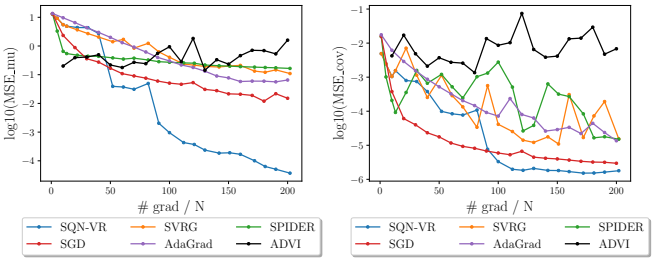
(a) N = 768, D = 9, cond(Σ) = 5, pima



(b) N = 1151, D = 20, cond(Σ) = 1106, diabetic



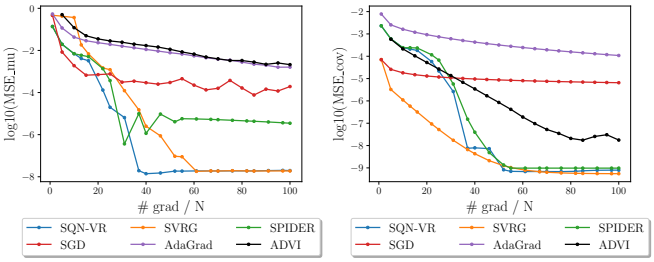
(c) N = 14980, D = 15, cond(Σ) = 4730, eeg



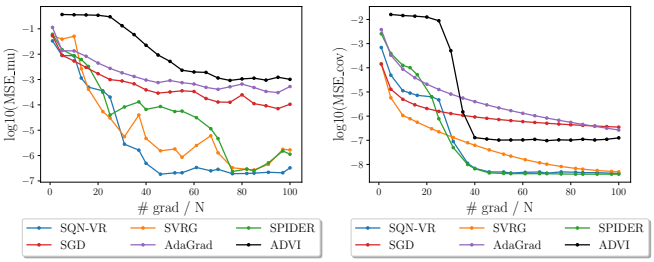
(d) N = 58000, D = 10, cond(Σ) = 3109, space

Figure A.12. Comparison with ADVI for Bayesian logistic regression (page 1)

Comparison with ADVI



(e) N = 100000, D = 19, cond(Σ) = 2412, susy



(f) N = 12214, D = 51, cond(Σ) = 58, mnist

Figure A.13. Comparison with ADVI for Bayesian logistic regression (page 2)